

МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Казанский национальный исследовательский технологический университет»**

**(ФГБОУ ВО «КНИТУ»)**

Кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Группа\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Уровень образования\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(бакалавр, специалист, магистр)

Вид ВКР\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(проектный, исследовательский, комбинированный)

Тема\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рецензент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Нормоконтролер\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Консультанты:

по экономической части\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

по безопасности и экологии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

по автоматизации производства

и метрологии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

по патентной части\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

по библиографии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Казань 2017

[**Введение** 5](#_Toc484969427)

[**1.** **История случайных чисел** 7](#_Toc484969428)

[1.1. Древний мир. 7](#_Toc484969429)

[1.2. Случайность в политике. 8](#_Toc484969430)

[1.3. Новое время. 8](#_Toc484969431)

[1.4. Новейшее время. 9](#_Toc484969432)

[1.5. Случайные и псевдослучайные числа 10](#_Toc484969433)

[1.6. Генераторы случайных чисел. 11](#_Toc484969434)

[1.7. Способность человека генерировать случайные числа 14](#_Toc484969435)

[1.8. Биологические датчики случайных чисел 14](#_Toc484969436)

[1.9. Генерация случайных чисел, основанная на движении мыши 15](#_Toc484969437)

[1.10. Генерация случайных чисел, основанная на скорости нажатия клавиш и кнопки. 16](#_Toc484969438)

[1.10. Оценка случайности чисел 17](#_Toc484969439)

[**2. Разработка программы для генерации случайных чисел с помощью биометрических параметров.** 19](#_Toc484969440)

[2.1. Выбор языка программирования для написания программы 19](#_Toc484969442)

[2.2. Архитектура программы 19](#_Toc484969443)

[2.3. Класс-форма MainForm 20](#_Toc484969444)

[2.4. Класс-форма RNGBUTTONSForm 22](#_Toc484969445)

[2.5. Класс-форма BRNGKeyForm 23](#_Toc484969446)

[2.6. Класс-форма BRNGMOUSEForm. 26](#_Toc484969447)

[2.7. Класс-форма BRNGMOUSEForm. 27](#_Toc484969448)

[**Заключение** 28](#_Toc484969449)

[**Cписок источников** 29](#_Toc484969450)

[**Приложения** 30](#_Toc484969451)

# **Введение**

Современный мир завязан на случайных числах. Они используются во многих сферах, таких как: наука, игры, статистика, криптография, искусство. Например, ученые проводят рандомизированные контролируемые испытания - это такой тип научного (часто медицинского) эксперимента, при котором его участники случайным образам делятся на группы, в одной из которых проводится исследуемое вмешательство, а в другой (контрольной) применяются стандартные методики или плацебо, а генераторы случайных и псевдослучайных чисел используются во многих компьютерных играх.

Для каждой области использования случайных чисел существуют определённые требования, удовлетворить которым могут специально подобранные методы. С точки зрения математики следует разделять случайные и псевдослучайные числа и способы их генерации, а значит, и области применения. Например, применение в криптографии, как правило, имеет жесткие требования, в то время как в других видах использования, где случайность не так важна, можно использовать более свободный стандарт псевдослучайности.

Одним из подходов к генерации последовательности чисел является использование биологических генераторов случайных чисел. Главной его особенностью является использование случайности биометрических данных (после их приведения в числовую систему) как начальную последовательность для псевдослучайных генераторов или непосредственно как случайную последовательность чисел. Это может иметь практическую значимость для генерации «ключа сессии» в таких протоколах как SSL, SSH, PGP или SET. Так же биологический датчик может стать заменой для программных генераторов случайных чисел, основанных на системном времени или содержимом буфера, в виду большей устойчивости к атакам злоумышленников.

Генераторы случайных чисел могут реализованы программно, либо аппаратно. Программная реализация подразумевает использование математических алгоритмов, результатом работы которых является статистически независимая последовательность чисел, соответствующая равномерному распределению. Стоит отметить что такая последовательность является детерминированной, так как получена из начального значения по определенному алгоритму. Следовательно, любая реализация подобного генератора принадлежит к категории псевдослучайных из-за своей детерминированности.

В связи с вышеизложенным можно сказать, что выбранная нами тема является важной частью криптографии в целом и в подходах генерации случайных чисел, в частности.

Целью данной работы является программная реализация биологического генератора случайных чисел. В теоретической части нами будут рассмотрены такие понятия, как случайные и псевдослучайные числа, случайность, биологические датчики генерации случайных чисел и подходы к их реализации, а также алгоритмы, которые будут использованы при написании программы. применение этих чисел и методы их генерации.

Поставлены следующие задачи:

* Рассмотреть понятия случайных и псевдослучайных чисел.
* Изучить концепцию биологических датчиков случайных чисел.
* Реализовать программно биологический датчик случайных чисел.

Работа состоит из двух глав, введения, заключения, списка использованной литературы и приложения.

# **История случайных чисел**

## 1.1. Древний мир.

Развитие понятия случайности в ходе истории человечества было крайне неравномерным. Историки часто задаются вопросом, почему прогресс развития данного концепта был таким медленным, учитывая, что люди сталкивались с ним испокон веков? Некоторые ученые считают, что среднестатистический человек той эпохи не был заинтересован в изучении и осмыслении случайности.

В древней истории концепция случайности и «шанса» были тесно завязаны на судьбе. В ту эпоху многие народы бросали гадальные кости чтобы предсказать судьбу, позже это переросло в разнообразные азартные игры. Свидетельства распространения таких игр находят в древнем Египте, Индии и Китае.

По всей видимости, первыми, кто формализовал случайности, были китайцы. Так, в одном из наиболее ранних китайских философских трудов «Книга перемен» рассматривалось гадание при помощи довольно сложных псевдослучайных процедур, использующих монеты или веточки тысячелистника, шести черт и поиска в книге соответствующей гексаграммы. Два основополагающих элемента китайской философии, инь и ян, были в «Книге перемен» скомбинированы в различных формах, чтобы продуцировать знакомую всем игру «*орел или решка».*

На Западе же, первыми, кто описывал случайность, были греки. Около 400 века до н.э греческий философ Демокрит представил свое предположение о мире, управляемом единообразующим законом мироздания, а случайность определил, как неспособность человека понять истинную природу некоторых событий. **[1]**

Так или иначе, до 16 века мир был сфокусирован на нематематическом аспекте случайности.

## 1.2. Случайность в политике.

Случайность имела и продолжает иметь место и в политической жизни человечества. Древнегреческая демократия в своем первоначальном состоянии предполагала власть народа, и многие руководящие должности выбирались случайным образом при помощи жребия. Такое распределение должностей позволяло избежать коррупции и концентрации власти у одной определенной ячейки общества.

В настоящее время выборы при помощи жребия ограничены, и используются главным образом в отборе [присяжных заседателей](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D1%81%D1%8F%D0%B6%D0%BD%D1%8B%D0%B9_%D0%B7%D0%B0%D1%81%D0%B5%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C) в англо-саксонской правовой системе в [Великобритании](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BB%D0%B8%D0%BA%D0%BE%D0%B1%D1%80%D0%B8%D1%82%D0%B0%D0%BD%D0%B8%D1%8F) и [США](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B5%D0%B4%D0%B8%D0%BD%D1%91%D0%BD%D0%BD%D1%8B%D0%B5_%D0%A8%D1%82%D0%B0%D1%82%D1%8B_%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B8). Некоторые страны, например, Ирак, тоже пытаются перенять данную практику, так как это положительно влияет на беспристрастность присяжных.

## 1.3. Новое время.

Примерно в 1620 году Галилео Галилей написал «Рассуждение об игре в кости», где он довольно подробно проанализировал эту задачу. В 1654 году, сподвигнутые интересом Антуана Гомбо к азартным играм, Блез Паскаль и Пьер Ферма проделали большую работу, чтобы положить основы для развития теории вероятности. «Пари Паскаля» отмечается как раннее использование понятия бесконечности и первое формальное использование теории вероятности. Работа Паскаля и Ферма повлияли в свою очередь на работу Лейбница и его «математический анализ», который в свою очередь способствовал дальнейшему пониманию вероятности и случайности.

В то время как математические гении той эпохи закладывали фундамент в современное понимание случайности, общество в своем большинстве продолжало полагаться на случайность во всевозможных предсказаниях и гаданиях.

Термин энтропия был предложен великим немецким ученым Рудольфом Клаузиусом в его труде «О различных удобных для применения формах второго начала математической теории теплоты», в котором было введено важнейшее для понимания случайности понятие [энтропии](https://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D1%8F).

## 1.4. Новейшее время.

В течение 20 века четыре главные интерпретации вероятности были хорошо изучены и поняты (рис. 1):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Классическая** | **Частотная** | **Субъективная** | **Свойственная** |
| **Главная гипотеза** | Принцип равноценности | Частота появления | Степень доверия | Степень детерминированных связей |
| **Концептуальный базис** | Гипотетическая симметрия | Предыдущие данные о системе | Знания и интуиция | Нынешнее состояние системы |
| **Концептуальный подход** | Гипотетический | Эмпирический | Субъективный | Метафизический |
| **Проблемы** | Противоречия в принципе равноценности | Цикличные определения | Проблема назначения классов | Спорная концепция |

Рис. 1. Классификация интерпретаций вероятности.

В это время случайные числа и случайность нашли применение во многих прикладных сферах, таких как финансы и физика. В 1900 году французский математик Л.Башелье при помощи Броуновского движения попытался описать финансовые опционы, что послужило основой развития финансовой математики и стохастических процессов.

Развитие квантовой механики в начале 20го века и формулирование принципа неопределённости Гейзенбе́рга в 1927 году положили конец Ньютоновскому мышлению среди физиков относительно определённости природы.

В 1948 году Клод Шеннон разработал информационную теорию, которая дала толчок энтропическому взгляду на случайность.

## 1.5. Случайные и псевдослучайные числа

  Случайные числа – это такая последовательность чисел, в которой невозможно предугадать следующее число, даже зная всю числовую последовательность. Псевдослучайные числа – это последовательность чисел, имеющая признаки случайных чисел, но каждое следующее число вычисляется по специально подобранному алгоритму. Несомненным преимуществом псевдослучайных чисел является их быстрая генерация и отсутствие необходимости в специальных аппаратных устройствах, но есть и существенный недостаток – запас псевдослучайных чисел ограничен.

В свою очередь, алгоритм, результатом выполнения которого является последовательность псевдослучайных чисел, называется генератором псевдослучайных чисел.

Стоит отметить, что имеет смысл генерировать псевдослучайные числа только с равномерным законом распределения, так как все остальные можно получить из равномерного распределения путём преобразований, известных из теории вероятности.

В свою очередь, равномерным называется такое распределение, при котором случайная величина имеет конечное количество равновероятных значений. Существует классический пример из теории вероятности. Пусть дана последовательность, состоящая из нулей и единиц, вероятность появления нуля в данной последовательности в среднем будет равна 50%. Но это не означает, что в последовательности длиной в 1000 цифр будет 500 нулей. Более того, в такой последовательности может быть 999 единиц, а вероятность того, что тысячным элементом будет нуль, так же составляет 50%. При рассмотрении же достаточно большой последовательности нули будут встречаться в половине случаев. Это утверждение кажется большинству парадоксальным, однако оно не противоречит реальности.

## 1.6. Генераторы случайных чисел.

В те времена, когда еще не были изобретены компьютеры, случайные числа получали довольно примитивным способом, вытягивая карты, вытаскивая разноцветные шары из мешков, бросая кости. Совершенно очевидно, что данные методы были далеки от совершенства, поэтому в 1927 году Л.Х.К. Типпетт разработал и опубликовал первую таблицу случайных чисел.

С развитием науки и технологий количество необходимых случайных чисел росло, тогда стали появляться машины, генерирующие случайные числа. Первая такая машина была использованная в 1939 году М.Ж. Кендаллом и Б. Бабингтон-Смитом для построения таблицы, содержащей 100 000 случайных чисел. Корпорация RAND при помощи таких устройств сгенерировала и опубликовала таблицу, состоящую из миллиона случайных чисел. Так же стоит упомянуть генератор случайных чисел ERNIE, применявшийся в течении многих лет в Великобритании для генерации номеров выигравших билетов лотереи. В наши дни, эти устройства тоже имеют применение, являясь генераторами энтропии. Такие генераторы способны генерировать истинные случайные числа, но у них есть существенный недостаток – цена. Именно поэтому и возникла необходимость в генераторах псевдослучайных чисел. **[2]**

После изобретения компьютеров начались исследования эффективного способа получения случайных чисел, встроенных программно в компьютеры. В начале применение таблиц было малоэффективным, так как из-за ограниченности памяти на заре появления компьютеров и необходимости ручного ввода таблицы не могли быть слишком большими. Технологический прогресс позволил вернуться к использованию таблиц в 90-е годы, миллиарды тестированных случайных байтов можно было разместить на таких носителях, как компакт диск. В 1995 году Дж. Марсалья подготовил демонстрационный диск с 650 мегабайтами случайных чисел, при генерировании которых запись шума диодной цепи сочеталась с определенно скомпонованной музыкой в стиле рэп. Сам Марсалья называл это белым и черным шумом. **[3]**

Несовершенство первых механических методов побудило к созданию генераторов псевдослучайных чисел при помощи арифметических операций, заложенных в компьютер. Одним из первых, около 1949 года, такой генератор предложил Джон фон Нейман. Данный метод получил название метода середины квадрата, его суть заключается в следующем: нужно возвести предыдущее случайное число в квадрат, а затем выделить средние цифры. Данный метод долго использовался для генерации случайных чисел, несмотря на то, что к нему было множество претензий.

Так же свой метод предлагал и Дональд Кнут, назвав его «супергенератор случайных чисел», его суть можно описать следующей последовательностью шагов:

К1. [Выбрать число итераций.] Присвоить Y наибольшую значащую цифру Х. (Мы выполним шаги К2-К13 точно Y+1 раз, т. е. применим рандомизированные преобразования случайное число раз.)

К2. [Выбрать случайный шаг] Присвоить следующую наибольшую значащую цифру X. Переходим к шагу К (3 + Z), т. е. к случайно выбранному шагу в программе.

КЗ. [Обеспечить> 5 х 109] Если X <5000000000, присвоить X значение X + 5000000000.

К4. [Средина квадрата.] Заменить X серединой квадрата X.

К5. [Умножить.] Заменить X числом (1001001001 X) mod 1010.

К6. [Псевдодополнение.] Если X <100000000, то присвоить X значение X + 9814055677; иначе присвоить X значение 1010- X.

К7. [Переставить половины.] Поменять местами пять младших по порядку знаков со старшими.

К8. [Умножить.] Выполнить шаг К5.

К9. [Уменьшить цифры.] Уменьшить каждую не равную нулю цифру десятичного представления числа X на единицу.

К10. [Модифицировать на 99999.] Если А' <105, присвоить X значение — X 2 +99999; иначе присвоить X значение X — 99999.

К11. [Нормировать.] (На этом шаге А' не может быть равным нулю.) Если X <109, то умножить X на 10.

К12. [Модификация метода средин квадратов.] Заменить Х на средние 10 цифр числа Х (Х — 1).

К13. [Повторить?] Если Y> 0, уменьшить У на 1 и возвратиться к шагу К2. Если Y = 0, алгоритм завершен. Значение числа X, полученное на предыдущем шаге, и будет желаемым «случайным» значением. **[4]**

После рассмотрения всех преобразований алгоритма К кажется правдоподобным, что можно было бы обеспечить бесконечное количество случайных числе. На самом деле, когда этот алгоритм впервые был реализован на компьютере, он почти мгновенно сошелся к 10-значному числу 6065038420, которое преобразовывалось само в себя.

Так же стоит упомянуть линейный конгруэнтный метод. Данную схему предложил Д. Г. Лехмер в 1949 году. Суть этого метода описывается формулой: **[5]**

**Ki = (A \* Ki-1 + C) mod M, I ≤ 0.**

где Ki следующее случайное число;

Ki-1 -предыдущее псевдослучайное число;

A, C, M – некоторые константы.

Например, при M = 2, а Кi-1=A=C=4 получим последовательность 4, 10, 12, 26, 54…

Данный метод находит широкое применение в стандартных библиотеках многих компиляторов, и, хотя исследования показывают, что он способен генерировать хорошую последовательность псевдослучайных чисел, его криптографическая стойкость невелика.

## 1.7. Способность человека генерировать случайные числа

Нельзя говорить о случайных числах, не упомянуть о способности человека генерировать их.

Это доказанный факт, что последовательности сгенерированные человеком не удовлетворяют критериям полной случайности. Среди ученых нет общего консенсуса по этому вопросу, существуют работы доказывающие то что числа генерируемые людьми являются случайными, а так же работы опровергающие этот факт. [7-8]

Но как же люди генерируют случайные числа? Если попросить назвать человека ряд случайных на его взгляд чисел, то каждый человек будет придерживаться определённой стратегии или шаблона, которые зависят от его представления о случайности.

Согласно исследованию У.Бейнса, генерация случайных чисел активирует определённые области в мозге человека. Если генерируемы числа не записываются, то это провоцирует их запоминание в так называемой оперативной памяти, что вызывает шаблонность генерации. [9]

Учитывая сказанное выше можно сделать вывод, что самостоятельно человек не способен генерировать случайные числа удовлетворяющие критериям случайности, но можно использовать данные полученные при помощи человека при генерации биологическими генераторами случайных чисел.

## 1.8. Биологические датчики случайных чисел

В данной работе будет рассмотрено и реализовано несколько алгоритмов, которые позволяют программно реализовать биологический датчик случайных чисел.

Смысл данного генератора состоит в использовании биологических параметров человека, после их преобразования в числовое представление. Источником энтропии в подобных генераторах могут выступать различные явления, не зависящие от человека напрямую такие как: скорость реакции пользователя на различные события, рисунок отпечатка пальца, радужки глаза, скорость расщепления определенных ферментов и т.п.

### 1.9. Генерация случайных чисел, основанная на движении мыши

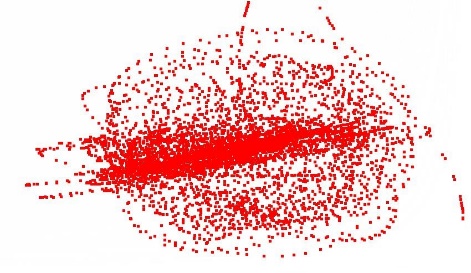
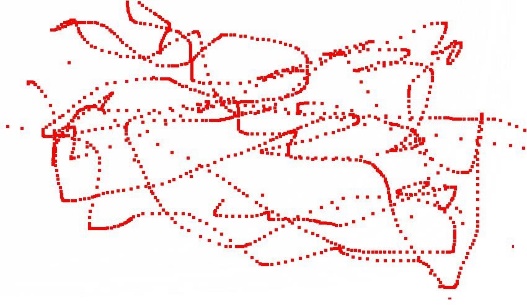
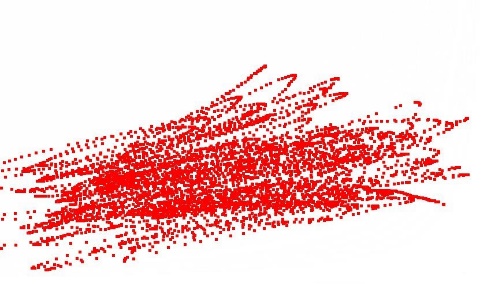
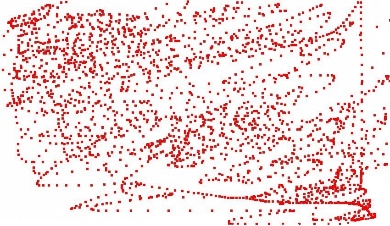
Как говорилось ранее, биологические генераторы случайных чисел позволяют получать «хорошие» последовательности. Что же может быть недетерминированным источником при реализации подобного генератора на персональном компьютере?

Существует множество подходящих явлений, таких как: температурный шум, атмосферный шум, период полураспада радиоактивных элементов или даже просто подбрасывание монетки. Все эти методы позволяют так или иначе получить случайные числа, но при попытке их использования на ПК возникает несколько проблем: эти методы либо слишком дороги для использования обычными пользователями, ведь мало кто имеет доступ к радиоактивным элементам, либо слишком медлительны, например, представьте сколько потребуется времени для генерации 1000 чисел при подбрасывании монетки. В большинстве случаев подобные недетерминированные источники требуют дополнительные аппаратные средства, что делает их использование не таким распространенным.

Другое дело компьютерная мышь, которая имеется практически у каждого пользователя персонального компьютера, скорость генерации чисел приемлема и не требует особых знаний или умений.

По своей сути движение мыши — это физическое движение и генерирует аналоговые сигналы, в то время как ПК работает с цифровыми. В связи с этим необходима конвертация аналоговых сигналов в цифровые, к счастью, практически все операционные системы уже сделали это за нас. При вызове имеющихся API (application programming interface), которые реализованы для каждой из популярных операционных систем, мы можем получать координаты движения мыши. Значения этих координат обычно находятся в диапазоне от одного до нескольких тысяч.

При использовании движения мыши как источника энтропии стоит отметить, что существуют определенные шаблоны движения мыши для одного пользователя. Другими словами, существует некоторая закономерность движения мыши в разные моменты времени для одного и того же пользователя. Соответственно разные пользователи могут иметь разные шаблоны. (Рис. 2)



(Рис. 2) Модель движения мыши различными пользователями.

В связи с похожестью разных вводов от одного и того же пользователя, мы должны внимательно обрабатывать эти данные. Другими словами, сгенерированная последовательность может быть так же похожа для одного и того же пользователя. Чтобы этого избежать, нужно сделать так, чтобы результат после обработки входных данных был очень чувствителен к их изменению. Например, можно брать координаты курсора в определённый момент времени и, если их произведение четное брать 0, а в случае нечетного 1.

### 1.10. Генерация случайных чисел, основанная на скорости нажатия клавиш и кнопки.

Стоит так же упомянуть о другом подходе к генерации случайных чисел при помощи биометрических данных, таком как интервалы реакции человека на какие-либо события, будь то мигание лампочки, появления кнопки и многое другое.

Если попросить человека сгенерировать случайные числа вербально, то, как уже было сказано, задействуется так называемая оперативная память, которая приводит к шаблонности генерируемых чисел. В то же время при печатании на клавиатуре происходит так называемое сенсомоторное взаимодействие. Стоит так же отметить что каждый человек имеет свой индивидуальный стиль и ритм печатания на клавиатуре, а временные интервалы являются нормально распределёнными.

Учитывая это можно сказать, что интервалы времени реакции человека на определенные события является неплохим источником энтропии. По аналогии с генерацией мышью этот способ так же не требует дополнительных аппаратных средств, прост в использовании и легок в реализации.

Мы можем с лёгкостью измерить время между кликами и нажатиями на клавиши с помощью встроенных API, но можем ли мы доверять этому источнику энтропии? Да, и вот почему:

* Мы уверены, что эти промежутки не зависят от пользователя, т.к. человек не способен наживать одинаково на одну и тоже кнопку.
* Данный источник энтропии сложно отследить для того, кто хочет «взломать» случайную последовательность. Злоумышленнику потребуется отследить сами нажатия, либо послушать их с помощью микрофона, что практически невозможно при использовании стандартного аудио оборудования, которым обладает пользователь.

Если делать вывод, то можно сказать, что временные интервалы между нажатиями на клавиши или кнопки можно назвать хорошим источником энтропии.

### 1.10. Оценка случайности чисел

Но как же определить, можно ли назвать сгенерированную последовательность случайной? На сегодняшний день существует множество подходов к оценке случайности последовательности чисел. В целом оценка сводится к определению наличия равномерного распределения, большого периода, равной частоты появления одинаковых строк. Еще в 1995 году было доказано, что любая последовательность, полученная в ходе работы какого-либо алгоритма, является не совершенной. [10] Это значит, что истинно случайная последовательность не может быть вычислена по какому-то ни было алгоритму, и должна быть непериодической и бесконечной.

# **2. Разработка программы для генерации случайных чисел с помощью биометрических параметров.**



## Выбор языка программирования для написания программы

На сегодняшний день, существует множество языков программирования, каждый из которых придерживается определенной парадигмы. Из всего спектра парадигм мы бы хотели выделить объектно-ориентированное программирование, которое позволяет не задумываться над низкоуровневыми процессами и сосредоточиться на архитектуре и классах.

Среди языков, которые реализуют этот подход к программированию, можно выделить такие языки как: Java, C#, PHP, Python, VB.NET, Objective-C и многие другие. Поэтому выбирая язык для реализации биологического датчика случайных чисел, были учтены следующие критерии: язык

1. Должен быть объектно-ориентированным;
2. Должен быть прост в изучении;
3. Должен иметь удобную среду разработки;
4. Должен компилироваться на ОС Windows;
5. Должен предоставлять удобные инструменты для реализации программы.

Исходя из всего этого, был выбран язык программирования C#, который является мощным ООП языком, разработанным Microsoft в начале 2000-х годов для разработки на платформе .Net. [6]

Выбранный язык имеет легкий синтаксис, но несмотря на это является мощным инструментом для реализации desktop приложений на платформе Windows. Как часть .Net платформы C# имеет доступ к широкому набору библиотек и API реализованных как самими Microsoft, так и сторонними разработчиками.

## Архитектура программы

Программа реализована как windows forms приложение и по существу разбита на несколько классов-форм.

1. **MainForm** – это стартовая форма, позволяющая осуществлять навигацию по приложению;
2. **RNGBUTTONSForm** – это форма для генерации случайных чисел при нажатии на кнопку;
3. **RNGKEYForm** – это форма для генерации случайных чисел по нажатию определённых клавиш на клавиатуре;
4. **RNGMOUSEForm** – это форма для генерации случайных чисел при передвижении курсора мыши в определенной области.
5. **BRNGStatistics –** это форма для отслеживания статистики по ходу генерации.

Каждая форма, отвечающая за генерацию случайных числе, реализует определенный алгоритм биологической генерации случайных чисел.

## Класс-форма MainForm

Данная форма является основой программы, на ней происходит выбор способа генерации, количество генерируемых чисел, а также можно увидеть уже сгенерированные числа и сохранить их в отдельный текстовый файл.

Для реализации этих функций на форме есть четыре элемента, три кнопки для перехода между формами на которых реализованы различные алгоритмы генерации чисел, один «TextBox» для хранения и представления сгенерированных чисел, кнопка «Сохранить» чтобы сохранить полученные числа, и поле для ввода длинны последовательности. (см. Приложение А).

При старте программы, происходит инициализация всех компонентов формы.

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

Инициализация позволяет вывести на экран главную форму программы и вывести все компоненты, такие как кнопки, текст боксы и прочее, а так же привязать к этим компонентам обработчики различных событий.

Теперь рассмотрим методы, содержащиеся в данном классе. Вот первый из них:

private void GenerateByButtonsButton\_Click(object sender, EventArgs e)

{

this.Hide();

BRNGForm numberGenerator = new BRNGForm(this);

numberGenerator.Show();

}

Данный метод вызывается при нажатии на кнопку «**Генерация по кнопке**» в **MainForm.** В этом методе создается экземпляр класса **BRNGForm**, отправляя как параметр конструктора экземпляр **MainForm** для того, чтобы можно было обратно вывести эту же форму при завершении генерации. Далее вызывается метод **Show** на этом экземпляре, который является наследуемым от базового класса **Form**.

Следующий метод похож на предыдущий, его задача та же, вывести нужный биологический генератор случайных чисел.

private void GenerateByLettersButton\_Click(object sender, EventArgs e)

{

this.Hide();

BRNG.Forms.BRNGKeyForm numberGeneratorByKeys = new Forms.BRNGKeyForm(this);

numberGeneratorByKeys.Show();

}

Данный метод вызывается при нажатии на кнопку «**Генерация по клавише**» в **MainForm.** В этом методе создается экземпляр класса **BRNGKeyForm**, отправляя как параметр конструктора экземпляр **MainForm** для того, чтобы можно было обратно вывести эту же форму, и вызывает метод **Show** на это экземпляре, который является наследуемым от базового класса **Form**.

Это метод работает по аналогии с двумя предыдущими, и является обработчиком нажатия на кнопку **«Генерация мышью»**.

private void GenerateByMouseButton\_Click(object sender, EventArgs e)

{

this.Hide();

BRNG.Forms.BRNGMOUSEForm numberGeneratorByMouse = new Forms.BRNGMOUSEForm(this);

numberGeneratorByMouse.Show();

}

Последний обработчик присутствующий на этой форме вызывается при нажатии на кнопку **«Сохранить»**, он вызывает **SaveFileDialog** предлагающий сохранить сгенерированные числа в текстовый файл. Это позволяет проанализировать полученную последовательность и использовать ее в дальнейшем.

private void saveButton\_Click(object sender, EventArgs e)

{

if (!string.IsNullOrEmpty(mainRichTextBox.Text))

{

Stream myStream;

SaveFileDialog save = new SaveFileDialog();

save.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

save.FileName = "myRandomNumbers.txt";

save.FilterIndex = 2;

save.RestoreDirectory = true;

if (save.ShowDialog() == DialogResult.OK)

{

if ((myStream = save.OpenFile()) != null)

{

using (var writer = new StreamWriter(myStream))

{

writer.Write(mainRichTextBox.Text);

}

myStream.Close();

}

}

}

}

## Класс-форма RNGBUTTONSForm

Данный класс-форма вызывается при нажатии на кнопку **«Генерация по кнопке»** на главной форме. На форме присутствует одна кнопка, которая меняет свое местоположение на форме при нажатии на нее (см. Приложение Б).

Этот класс-форма реализует алгоритм биологического датчика при помощи интервалом между нажатиями на кнопку. Эти интервалы используются как источник энтропии и берутся они как промежутки между нажатиями на кнопку. Встроенный в .Net таймер позволяет отмерять время между нажатиями на кнопку.

Рассмотрим структуру данного класса подробнее.

public BRNGForm(MainForm form)

{

InitializeComponent();

entropy = new Stopwatch();

previousForm = form;

Point = new Engine.RandomPointGeneratorEngine();

entropy.Start();

}

В конструкторе мы инициализируем компоненты, присутствующие в окне данной формы, и объявим необходимые поля:

1. Экземпляр класса **RandomPointGeneratorEngine** **Point** для задания положения кнопки;
2. Экземпляр класса **Stopwatch** **Entropy** для отсчета времени между нажатиями на кнопку, которое и будет псевдослучайным числом;
3. Экземпляр класса **MainForm PreviousForm** для хранения предыдущей формы и возможности возврата к ней.

А также запустим наш таймер.

При загрузке формы выводится сообщение с помощью, которое описывает способ генерации, а также инициализируется и запускается форма, отвечающая за статистику. Статистика позволяет следить за количеством уже сгенерированных чисел.

private void BRNGForm\_Load(object sender, EventArgs e)

{

MessageBox.Show("Для генерации случайных чисел нажимайте на картинку самолета.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);

statistics = new BRNGStatistics((int)previousForm.lengthOfSeqBox.Value);

statistics.Show();

statistics.Left += this.Size.Width+statistics.Size.Width;

}

Данный метод **BRNGPicture\_Click** является обработчиком события при нажатии на кнопку с изображением самолета.

private void BRNGPicture\_Click(object sender, EventArgs e)

{

entropy.Stop(); previousForm.mainRichTextBox.AppendText((entropy.ElapsedMilliseconds).ToString() + " ");

statistics.LengthOfSeq--;

BRNGPicture.Location = Point.GenerateNewPoint(this.Size.Height, this.Size.Width);

entropy.Reset();

entropy.Start();

if (statistics.LengthOfSeq == 0)

{

this.Close();

statistics.Close();

statistics.Dispose();

previousForm.Show();

}

}

После нажатия на кнопку прекращается отсчет времени и проверяется условие. Если количество нажатий не превышает необходимое количество, тогда в **mainRichTextBox,** находящийся на главной форме, мы отправляем количество миллисекунд, прошедших между нажатиями на кнопку, сбрасываем секундомер **entropy** и запускаем его снова. Если же количество нажатий на кнопку не меньше необходимого количества, тогда мы закрываем эту форму и выводим главную.

private void BRNGForm\_FormClosed(object sender, FormClosedEventArgs e)

{

statistics.Close();

statistics.Dispose();

previousForm.Show();

}

Метод **BRNGForm\_FormClose** является обработчиком события закрытия формы **BRNGForm** и вызывается при ее закрытии пользователем, для того чтобы вывести предыдущую форму.

## Класс-форма BRNGKeyForm

Данный класс-форма вызывается при нажатии на кнопку **«Генерация по клавише»** на главной форме. На этом окне присутствует буква английского алфавита, которая меняет свое положение в окне после нажатия на соответствующую клавишу на клавиатуру, и после такого нажатия также меняется сама буква. (см. Приложение В).

public BRNGKeyForm(MainForm \_previusForm)

{

newChar = new Engine.RandomPointGeneratorEngine();

previousForm = \_previusForm;

InitializeComponent();

entropy = new System.Diagnostics.Stopwatch();

entropy.Start();

}

В конструкторе мы инициализируем компоненты, присутствующие в окне данной формы, и объявим необходимые поля:

1. Экземпляр класса **RandomPointGeneratorEngine** **newChar** для генерации новой буквы;
2. Экземпляр класса **Stopwatch** **Entropy** для отсчета времени между нажатиями на кнопку, **которое** и будет псевдослучайным числом;
3. Экземпляр класса **MainForm PreviousForm** для хранения предыдущей форма и возможности возврата к ней.

В качестве источника энтропии для этого вида генератора также, как и для предыдущего используются временные промежутки между нажатиями пользователя на клавишу, соответствующую букве на форме.

private void BRNG\_KeyPress(object sender, KeyPressEventArgs e)

{

if (string.Equals(keyFormLabel.Text, e.KeyChar.ToString(), StringComparison.OrdinalIgnoreCase))

{

statistics.LengthOfSeq = statistics.LengthOfSeq - 1;

entropy.Stop();

previousForm.mainRichTextBox.AppendText(entropy.ElapsedMilliseconds.ToString() + " ");

keyFormLabel.Text = newChar.GenerateNewChar().ToString();

entropy.Reset();

entropy.Start();

keyFormLabel.Location = newChar.GenerateNewPoint(this.Size.Height, this.Size.Width);

}

else

{

statistics.ErrorsCounter++;

statistics.LengthOfSeq = statistics.LengthOfSeq + 1;

if (statistics.ErrorsCounter > 5)

{

MessageBox.Show("Слишком много ошибочных нажатий, попробуйте еще раз!", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Error);

previousForm.mainRichTextBox.Clear();

previousForm.CheckResultOfGeneration(false);

previousForm.Show();

this.Close();

}

}

if (statistics.LengthOfSeq ==0 )

{

CustomClose(false);

}

}

Теперь рассмотрим сам метод генерации псевдослучайной последовательности. При загрузке формы в окне появляется буква, задача пользователя нажать соответствующую клавишу на клавиатуре. Метод **BRNG\_KeyPress** срабатывает при нажатии пользователем на любую клавишу клавиатуры, далее идет проверка соответствия нажатой клавиши запрашиваемому символу, и если все верно, то мы считаем данное нажатие успешным, а также останавливаем секундомер и отправляем на **MainForm** сгенерированное число. Далее идет перезапуск таймера и генерация нового местоположения для символа, так же, как и самого символа. Если же пользователь совершает ошибку, то увеличивается количество ошибок в форме статистики и добавляется количество нажатий.

При загрузке формы выводится сообщение с описание работы данного генератора, а также инициализируется форма статистики для слежения за процессом генерации случайных чисел.

private void BRNGKeyForm\_Load(object sender, EventArgs e)

{

MessageBox.Show("Нажимайте на клавиатуре клавиши соответстующие показанным на экране.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);

statistics = new BRNGStatistics((int)previousForm.lengthOfSeqBox.Value);

statistics.Show();

statistics.Left += this.Size.Width;

}

Метод **CustomClose** является обработчиком события закрытия формы **BRNGKeyForm** и вызывается при ее закрытии пользователем, для того чтобы вывести предыдущую форму.

private void CustomClose(bool closed)

{

if (!closed)

{

this.Close();

}

statistics.Close();

statistics.Dispose();

previousForm.Show();

}

Два предыдущих генератора случайных чисел реализовали биологические генераторы случайных чисел, энтропией в них являлись интервалы между какими-либо действиями, требующими участия пользователя. Т.к. пользователь не может полностью контролировать такие действия как ритм и скорость нажатия на кнопку, или реакцию на появление новой буквы на экране, то данный вид генераторов можно считать хорошими генераторами, а источник энтропии недетрменированным.

## Класс-форма BRNGMOUSEForm.

Данный класс-форма вызывается при нажатии на кнопку **«Генерация мышью»** на главной форме. На форме картинка-поле для генерации псевдослучайных чисел и шкала для отсчета сгенерированных чисел (см. Приложение Г).

Этот генератор случайных чисел в качестве источника энтропии использует физическое перемещение мыши. Которое так же сложно предсказать как и время реакции пользователя. Но, как уже было продемонстрированно в теоретической части, для одного и того же пользователя существует некий паттерн движений, и поэтому использование векторов движения мыши без их обработки, нельзя назвать надежным источником энтропии.

Исходя из вышеизложенного, был сделан вывод, что данный генератор будет обрабатывать входящие данные, и преобразовывать координаты движения мыши в двоичную величину. Для этого будет производиться несложная операция:

1. Берутся координаты мыши в определенный момент времени, т.е. неизвестно какие именно координаты берутся.
2. Координаты X и Y складываются и далее, если полученное число четное, то в результат записывается 0, а если нечётное, то 1.

Таким образом будет реализована одна из особенностей хеш функций, а именно, при небольшом изменении входных данных выходное значение изменяется сильно, а также зная выходное значение мы не сможем восстановить значения, от которых оно получилось.

Теперь подробно про программную реализацию.

public BRNGMOUSEForm(MainForm \_PreviusForm)

{

previusForm = \_PreviusForm;

InitializeComponent();

this.BRNGMouseFormProgressBar.Maximum = (int)previusForm.lengthOfSeqBox.Value;

controler = new Stopwatch();

controler.Start();

var pixel = new Bitmap(1, 1);

pixel.SetPixel(0, 0, Color.White);

saveTrace = new Bitmap(pixel, mouseFormPictureBox.Width, mouseFormPictureBox.Height);

trace = new List<PictureBox>();

}

В начале как обычно объявляем поля и инициализируем их в конструкторе.

Также в этой форме реализовано запоминание пути движения мыши пользователя, после анализа которых был сделан вывод о паттернах движения для одного и того же пользователя.

Теперь рассмотрим основной метод, отвечающий за генерацию случайных чисел.

private void mouseFormPictureBox\_MouseMove(object sender, MouseEventArgs e)

{

int cursorX = e.Location.X;

int cursorY = e.Location.Y;

ShowLocationOnScreen(cursorX, cursorY, false);

if (counter >= previusForm.lengthOfSeqBox.Value)

{

var path = Directory.GetCurrentDirectory();

path += "trace.jpeg";

saveTrace.Save(path, System.Drawing.Imaging.ImageFormat.Jpeg);

this.Close();

previusForm.Show();

}

if (controler.Elapsed.Milliseconds > 500)

{

counter++;

ShowLocationOnScreen(cursorX, cursorY, true);

if ((cursorX + cursorY) % 2 == 0)

{

previusForm.mainRichTextBox.AppendText((0).ToString() + " ");

}

else

{

previusForm.mainRichTextBox.AppendText((1).ToString() + " ");

}

controler.Reset();

controler.Start();

BRNGMouseFormProgressBar.Value = counter;

}

}

Сама генерация в данном способе происходит при движении мыши по выделенному полю. Метод **mouseFormPictureBox\_MouseMove** срабатывает только при передвижении мыши над изображением. Сам процесс генерации уже был описан в начале этого параграфа. Так как созданная программа предполагается для использования в учебных целях, то точки, которые берутся для генерации отмечаются красными точками, что позволяет обозреть работу алгоритма.

Полученный результат так же, как и предыдущие отправляется **MainForm**.

При загрузке этой формы появляется подсказка, объясняющая способ работы генерации.

private void BRNGMOUSEForm\_Load(object sender, EventArgs e)

{

MessageBox.Show("Для генерации случайных чисел перемещайте курсор мыши над картинкой.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

Также в этой форме есть метод позволяющий получить картинку, которая отображает путь движения мыши пользователя.

private void ShowLocationOnScreen(int cursorX, int cursorY, bool show)

{

locationOfMause = new PictureBox();

locationOfMause.Location = new Point(

cursorX,

cursorY

);

locationOfMause.Size = new System.Drawing.Size(5, 5);

locationOfMause.BackColor = Color.Red;

locationOfMause.Visible = show ? true : false;

locationOfMause.BringToFront();

locationOfMause.Name = "locaton" + cursorX.ToString();

locationOfMause.Parent = mouseFormPictureBox;

mouseFormPictureBox.SendToBack();

mouseFormPictureBox.Controls.Add(locationOfMause);

mouseFormPictureBox.Invalidate();

locationOfMause.DrawToBitmap(saveTrace, new Rectangle(new Point(cursorX, cursorY), new Size(5, 5)));

}

Для этого создается bitmap изображение, размером с картинку-поле на которой происходит генерация, и на ней ставятся точки, при каждом передвижении мыши, а после завершения генерации полученное изображение сохраняется как jpeg картинка.

## Класс-форма BRNGStatistics

Для отслеживания статистики генерации некоторые формы вызывают дополнительное окно статистики, которое реализовано в классе форме BRNGStatistics.

В конструкторе данной формы:

public BRNGStatistics(int \_lenghtOfSeq)

{

lenghtOfSeq = \_lenghtOfSeq;

statisticDatas = new BindingList<Data>();

InitializeComponent();

statisticDatas.Add(new Data { Value = lenghtOfSeq, Error = 0 });

dataGrid.DataSource = statisticDatas;

}

Инициализируется окно этой формы, состоящее из таблицы **DataGrid**, к которой привязано свойство **lengthOfSeq**, передающееся из вызывающей формы. Далее при изменении значения этого свойства, соответствующая строка в этой таблице будет принимать соответствующее значение. То же самое справедливо и для свойства отвечающее за ошибки.

public int ErrorsCounter

{

get { return errorCounter; }

set

{

errorCounter = value;

statisticDatas[0].Error = errorCounter;

dataGrid.Refresh();

}

}

public int LengthOfSeq

{

get { return lenghtOfSeq; }

set

{

lenghtOfSeq = value;

statisticDatas[0].Value = lenghtOfSeq;

//dataGrid.DataSource = statisticDatas;

dataGrid.Refresh();

}

}

# **Заключение**

В данной работе было рассмотрено понятие случайных и псевдослучайных чисел, была представлена история развития понятия случайности и его влияние на жизнь человека. Были рассмотрены генераторы случайных чисел, а в частности биологический генератор случайных чисел.

В практической части работы была разработана и реализована программа, реализующая некоторые из способов работы биологического датчика случайных чисел.

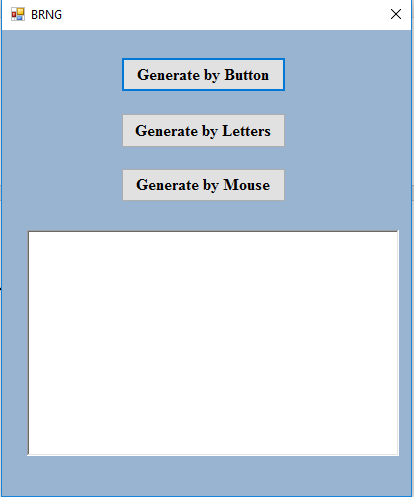
Подводя итог, хотелось бы сказать, что случайные числа играют важную роль в современном мире, они нашли свое применение во многих сферах и их генерация является насущной проблемой. Для решения данной проблемы были разработаны разнообразные аппаратные генераторы случайных чисел, но они не находят широкого применения из-за своей цены и необходимости встраивания в компьютер. С другой стороны, существуют генераторы псевдослучайных чисел, использующие разнообразные алгоритмы для генерации псевдослучайных последовательностей. Одним из них является биологический генератор случайных чисел. Он позволяет генерировать хорошие последовательности псевдослучайных чисел, но из-за времени генерации он не подходит для генерации большого количества чисел.

# **Cписок источников**

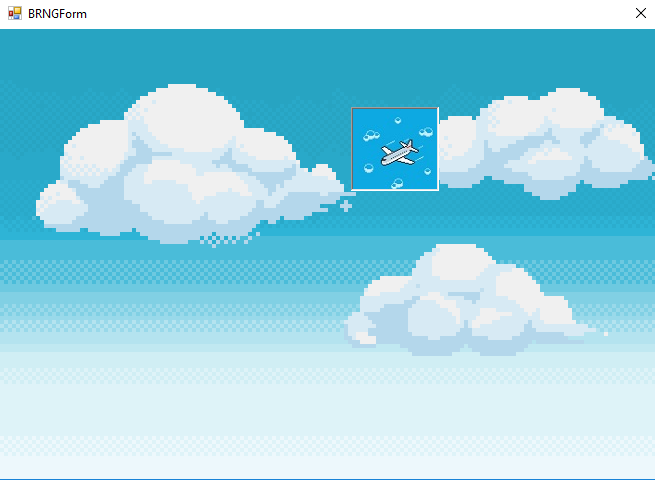
1. Брокгауз Ф.А., Ефрон И.А. Энциклопедический словарь Брокгауза и Ефрона в 82 тт. и 4 доп. тт. — М.: Терра, 2001 г. — 40 726 с.;
2. Harry F. ERNIE - A Random Number Generator [Электронный ресурс] - http://www.i-programmer.info/history/machines/6317-ernie-a-random-number-generator.html;
3. [Д. Кнут. Искусство программирования. Том 3](http://computersbooks.net/index.php?id1=4&category=teoriyaprogramirovaniya&author=shtona-v&book=20012) — Вильямс, 2001. — 788 c;
4. Д. Кнут. Искусство программирования. Том 2. Получисленные алгоритмы. — М.: Вильямс, 2000 —788с.;
5. Линейный конгруэнтный метод [Электронный ресурс] - http://www.intuit.ru/studies/courses/691/547/lecture/12383?page=2;
6. Д. Рихтер. CLR via C# Программирование на платформе Microsoft .Net Framework 4.5 на языке C#. 4-е изд. – СПБ.: Питер, 2016. – 896с.
7. Humans cannot consciously generate random numbers sequences: Polemic study.
8. Humans can consciously generate random number sequences: a possible test for artificial intelligence.
9. Random number generation and creativity.
10. Herring, C., Palmore, J.I. Random Number Generators are Chaotic, 1995)

# **Приложения**

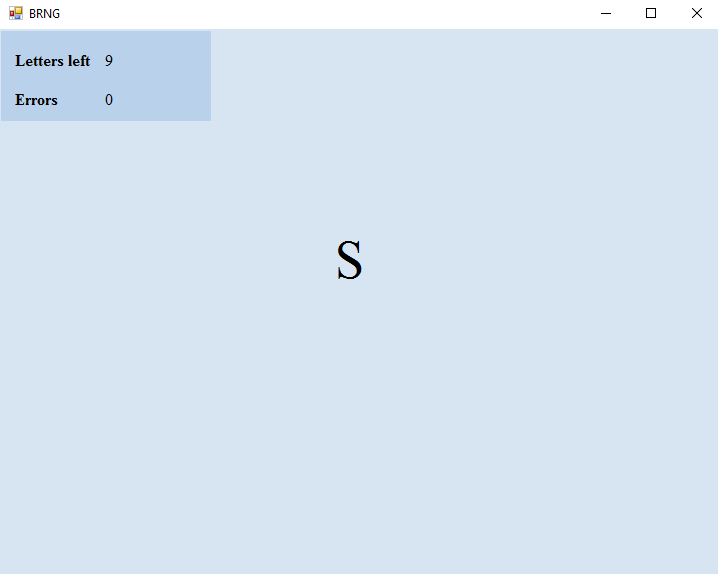
**Приложение А**



**Приложение Б**



**Приложение В**



**Приложение Г**

